DualMap: Enabling Both Cache Affinity and Load Balancing for Distributed LLM Serving

Ying Yuan[1] Pengfei Zuo[2,*] Bo Wang[2] Zhangyu Chen[2] Zhipeng Tan[1,*] Zhou Yu[2]

[1]Huazhong University of Science and Technology [2]Huawei

Motivation: why existing schedulers struggle

Two targets

1. Cache affinity

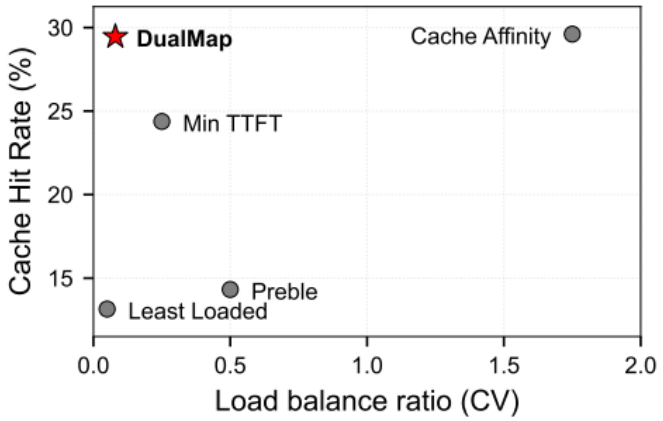
Co-locate shared prefixes to maximize KV-cache reuse and reduce TTFT.

2. Load balancing

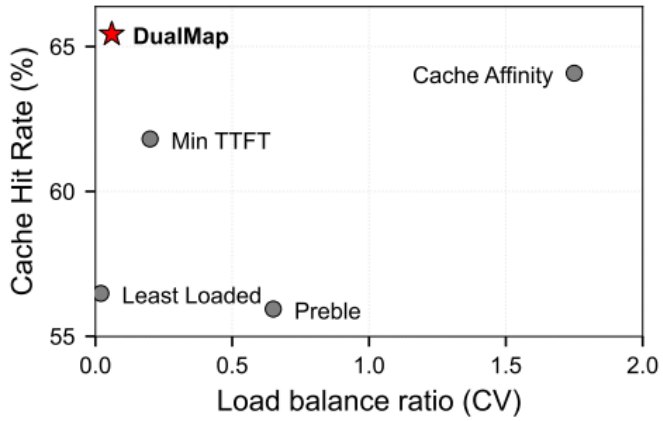
Spread requests evenly to avoid hotspots, queue buildup, and tail spikes.

Why this is hard

- A single mapping space usually favors one objective at a time.
- Hot-prefix routing overloads machines, while purely load-aware routing destroys locality.



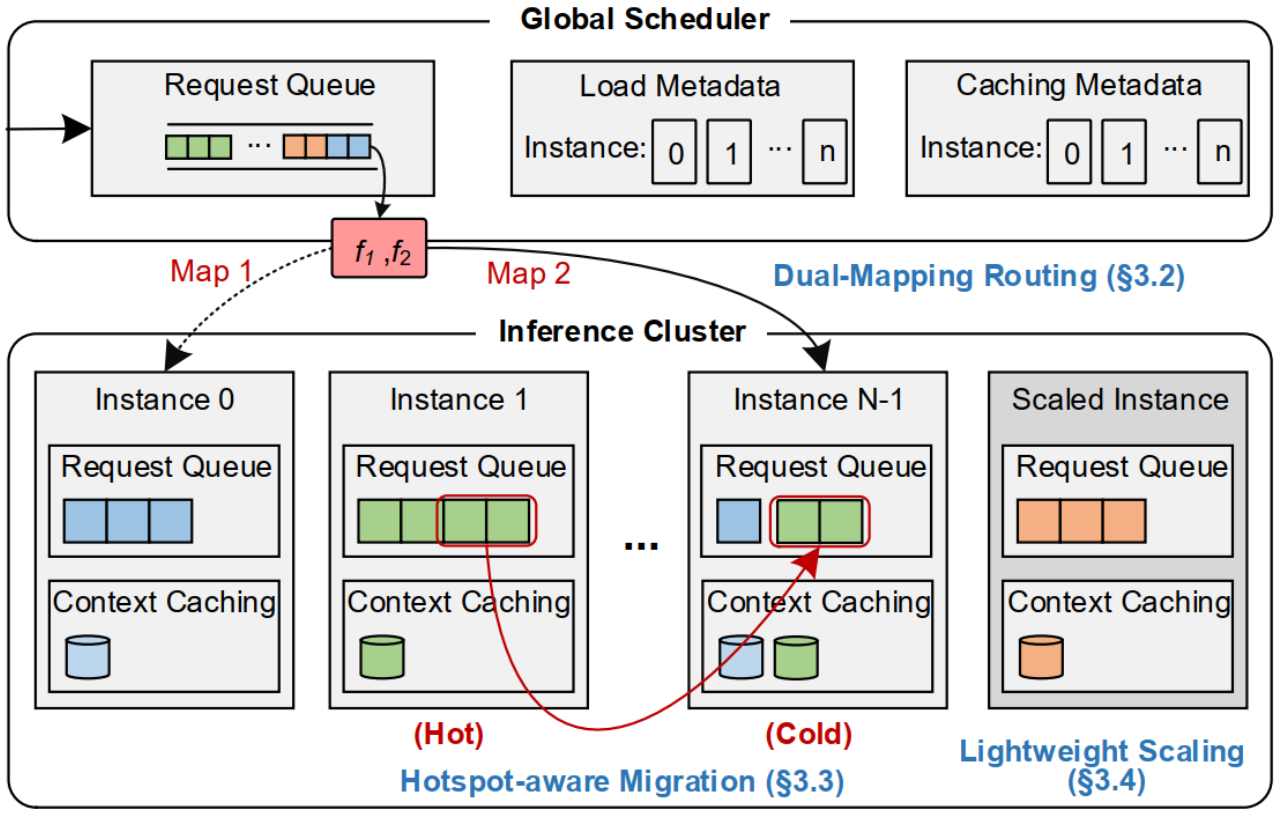
(a) Conversation



(b) Tool&Agent

Prior schedulers operate in one mapping space, so they can only trade one objective for the other.

DualMap overview



For each request prefix p, use two independent hash functions to obtain two candidate instances. Two choices are enough to preserve locality while improving balance.

Prefix(p) → { I_1 , I_2 }

Three techniques

- **SLO-aware routing**: prefer cache reuse, switch only when expected TTFT exceeds the SLO.
- **Hotspot-aware rebalancing**: migrate queued requests only within the same candidate pair.
- **Dual-hash-ring scaling**: support instance add/remove with only local remapping.

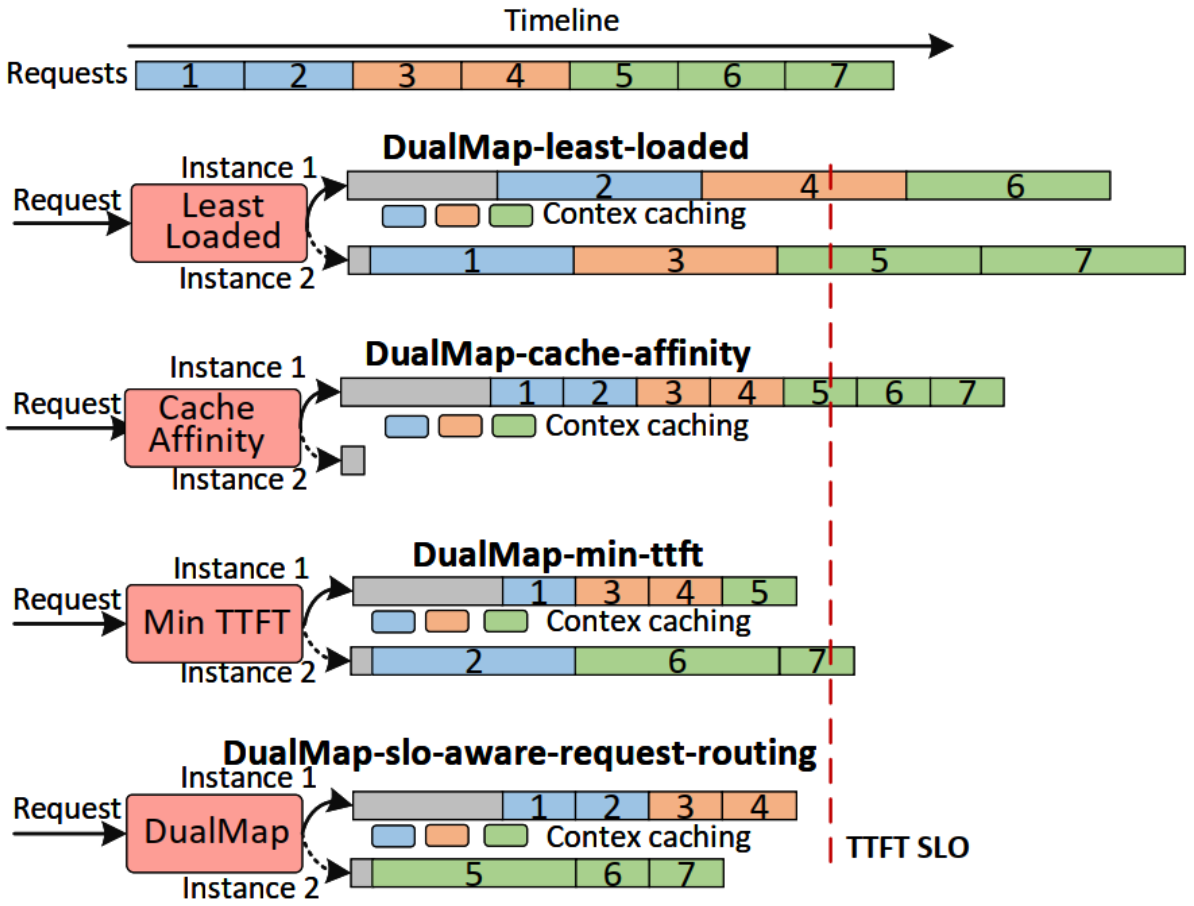
Technique 1: SLO-aware request routing

Keep sending requests to the cache-favorable instance until its expected TTFT would violate the SLO; then switch to the less-loaded candidate.

Cache first → SLO check
→Load-aware fallback

Why this matters

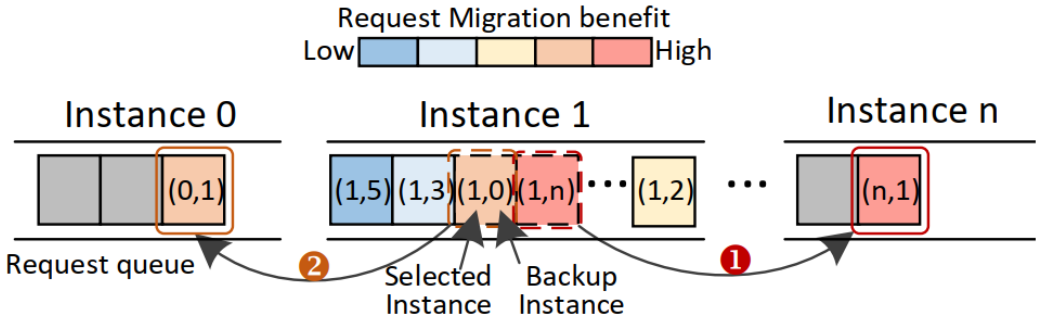
- Avoids min-TTFT oscillation under load fluctuation.
- Stabilizes cache hit rates instead of repeatedly bouncing between nodes.
- If both candidates have equal reuse, simply choose the less-loaded one.



Technique 2 & 3: hotspot handling and elastic scaling

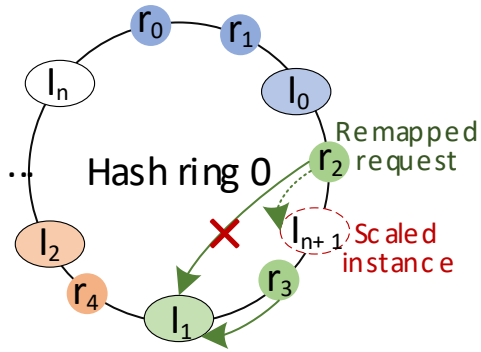
Hotspot-aware request rebalancing

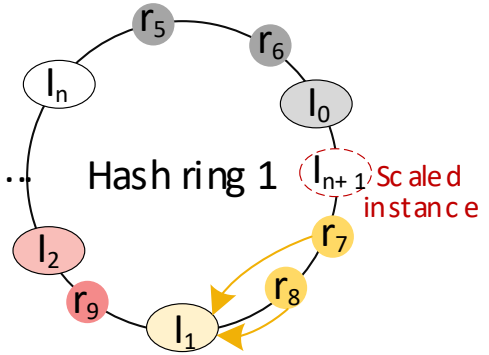
- Detect overloaded instances caused by skewed prefix popularity.
- Estimate the TTFT benefit of moving queued requests to the backup instance.
- Migrate only requests whose backup is underloaded and still SLO-safe.
- Rebalances only within the original candidate pair.



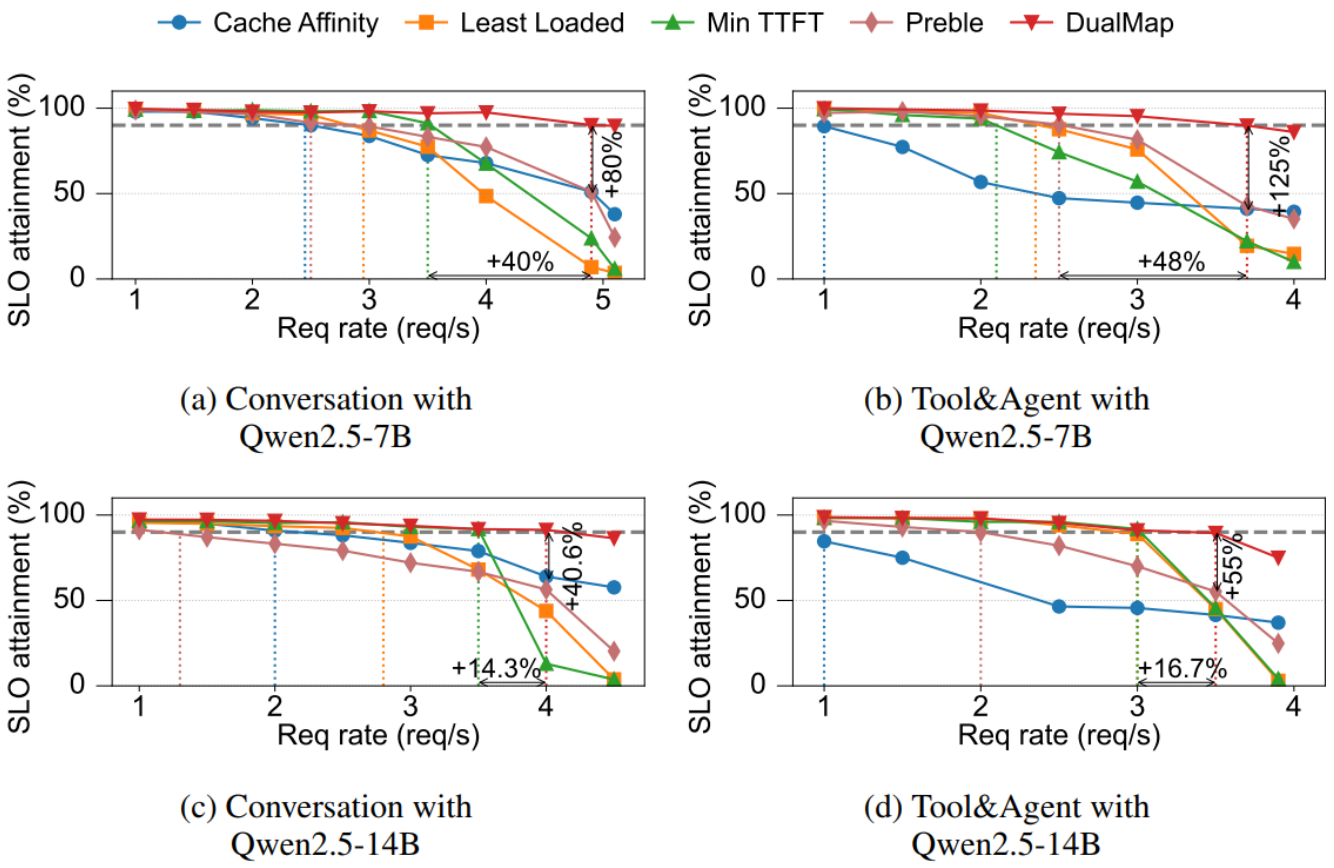
Lightweight dual-hash-ring scaling

Only local regions on the ring are remapped when instances are added or removed, so cache disruption stays small.





Main results: higher effective capacity and goodput



(a) Conversation with Qwen2.5-7B

(b) Tool&Agent with Qwen2.5-7B

(c) Conversation with Qwen2.5-14B

(d) Tool&Agent with Qwen2.5-14B

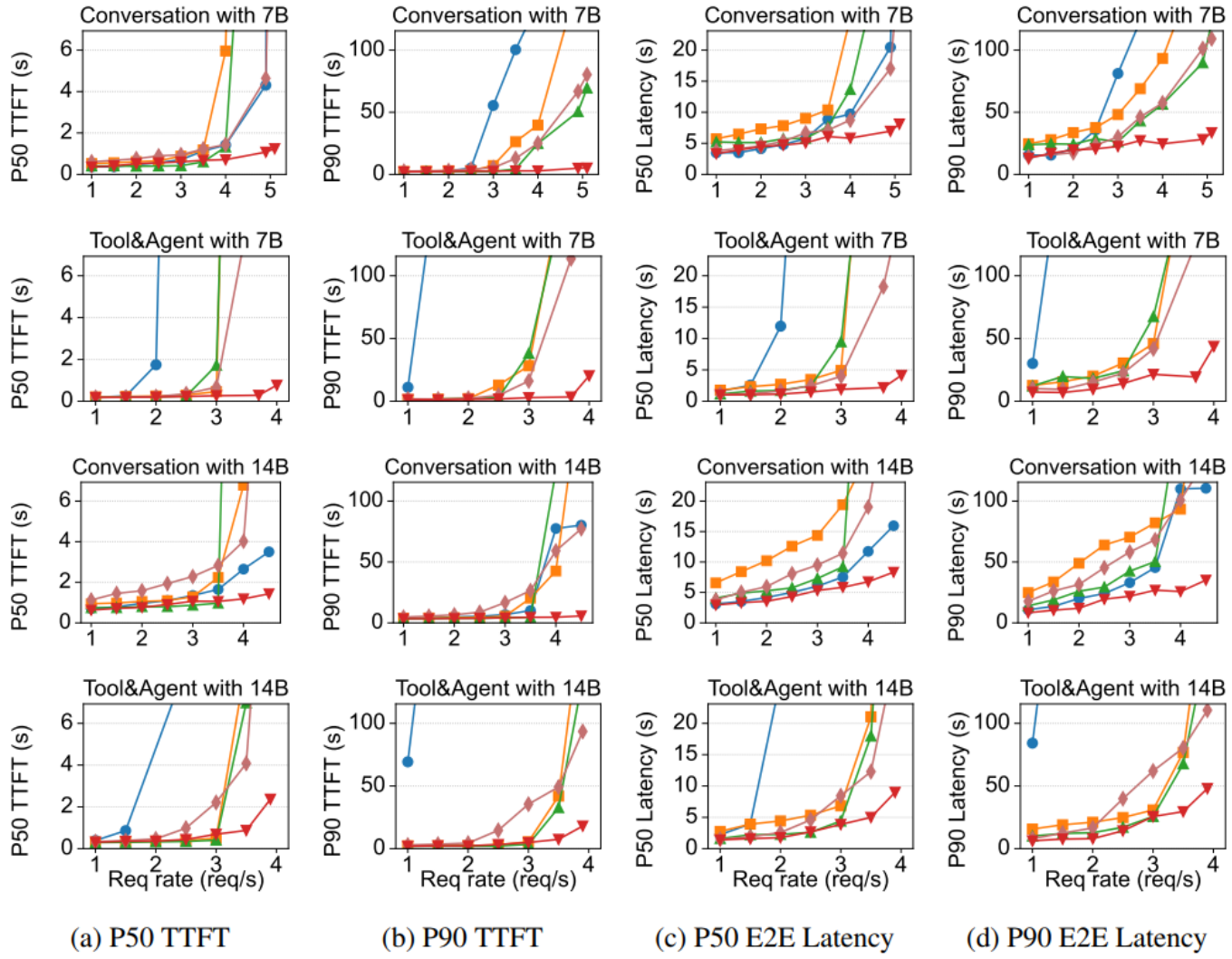
↑ 125% effective request capacity gain

↑ 16.7%–48% goodput improvement versus the best baseline

up to 2.25× overall effective request capacity under the same TTFT SLO

DualMap improves capacity because it reduces redundant prefill work while preventing queue buildup on hot instances.

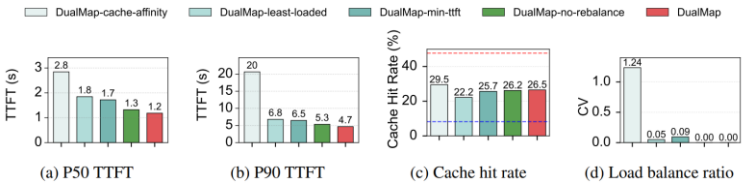
Why it works: latency drops and ablation evidence



Latency improvements

- P50 TTFT reduced by 55.4%–97.4% versus the best baseline.
- P90 TTFT reduced by 82.3%–97%, showing strong tail-latency control.
- Latency stays near low-QPS levels even when QPS doubles.

Ablation: both mechanisms matter



SLO-aware routing lowers TTFT; hotspot rebalancing gives an additional P90 TTFT reduction.

Takeaways

What to remember

1. The real breakthrough is dual mapping, not a better single-score scheduler.
2. DualMap keeps cache reuse whenever possible and spends load-balancing budget only when the SLO is at risk.
3. On real workloads, this translates into higher effective capacity, better goodput, and much lower tail TTFT.

Thank You !

Ying Yuan, yuanying@hust.edu.cn

Code & Paper